IN THE CLAIMS

Please amend the claims to the following.

1. (Currently Amended) A method, comprising:

encountering a non-privileged user-level programming instruction;

creating, responsive to the programming instruction, a first shared resource thread (shred) that is associated with a first private portion of a first application state and shares a second shared portion of the first application state with at least a second shred, wherein creating the first shred is performed without the intervention of an operating system; ~~belongs to the same process as one or more other shreds ;~~ and

executing, responsive to the programming instruction, the first shred concurrently with at least the second shred. ~~one of the one or more other shreds~~;

~~wherein creating the first shred is performed without the intervention of an operating system.~~

2. (Currently Amended) The method of claim 1, ~~further comprising: maintaining a private state for the first thread,~~ wherein the first private portion of an application state is associated with at least one of a plurality of registers including general purpose registers, floating point registers, MMX registers, segment registers, a flags register, an instruction pointer, control and status registers, SSE registers, and a MXCSR register.

3. (Currently Amended) The method of claim 1, ~~further comprising:~~ wherein the first application state is ~~sharing a state among a plurality of shared resource threads ("shreds")~~ associated with a first-operating-system-generated thread to execute a first process. ~~the plurality of shreds including the first shred and the one or more other shreds; while not sharing said state with a second shred, created as a result of a second non-privileged user-level programming instruction, that is associated with a second operating-system-generated thread~~.

4. (Currently Amended) The method of claim 1, further comprising:

creating a second operating system generated thread to execute a second process in response to a privileged non-user level programming instruction, the second operating system generated thread to be associated with a second application state; and

creating a third shred, in response to encountering a second non-privileged user-level programming instruction associated with the second process, wherein the third shred is to be associated with a shared portion and a private portion of the second application state,

wherein the second operating system generated thread and the third shred do not share a portion of the first application state.

~~sharing a state among a plurality of shreds of the one or more shreds; and~~

~~storing the state in one or more registers.~~

5. (Currently Amended) The method of claim 1, wherein the first shred and the ~~one or more shreds~~ second shred share a current privilege level and share a common address translation.

6. (Previously Presented) The method of claim 1, further comprising: receiving a non-privileged user-level programming instruction that encodes a shred.

7. (Currently Amended) The method of claim 1, further comprising communicating between the first shred and the second shred ~~at least one of the one or more other shreds~~.

8. (Currently Amended) The method of claim 1, further comprising sharing a system state among at least the ~~one or more~~ first and second shreds.

9. (Previously Presented) The method of claim 7, wherein said communicating is performed via one or more shared registers.

10. (Currently Amended) The method of claim 1, further comprising:
scheduling, responsive to a user-level programming instruction, the first shred and the second shred ~~one or more other shreds~~ without intervention of the operating system.

11. (Previously Presented) The method of claim 7, wherein said communicating is performed via a user-level shred signaling instruction.

12. (Currently Amended) The method of claim 11, further comprising:

storing at least the first private portion of the first application state in a memory ~~one or more shred states associated with the one or more shreds~~ responsive to receipt of a context switch request.

13. (Previously Presented) The method of claim 1, further comprising: handling with user-level exception handler code an exception generated during execution of the first shred, without intervention of the operating system.

14. (Previously Presented) The method of claim 13, further comprising: receiving the exception from an application program; and determining whether to report the exception to the operating system.

15. (Previously Presented) The method of claim 1, wherein the shred is to perform input/output (I/O) operations.

16. (Currently Amended) The method of claim 1, wherein the ~~one or more shreds~~ first and second shreds are to perform computation functions.

17. (Currently Amended) An apparatus, comprising:

execution resources to execute a plurality of instructions

the execution resources to receive a non-privileged user instruction;

the execution resources further to, responsive to the received instruction, execute a first shared resource thread ("shred") concurrently with at least a second shred ~~one or more other shreds~~; and

a shared register ~~that is~~ to be addressable by a user-level instruction, the shared register to be directly accessible by at least the first shred and the second shred to provide communication between the first shred and the second shred ~~one or more other shreds~~.

18. - 19. (Canceled).

20. (Currently Amended) The apparatus of claim 17 ~~18~~, wherein the shared register ~~one or more shared registers further comprise~~ comprises a first register ~~that enables~~ to enable an operating system or BIOS to enable multithreading architecture extensions for user-level multithreading.

21. (Currently Amended) The apparatus of claim 17, wherein the shared register comprises a first register to be atomically updated to synchronize data between the first and the second shred. ~~the execution resources are further to, responsive to the received instruction, begin execution of a shred concurrently with one or more other shreds, without control of an operating system~~.

22. (Previously Presented) The apparatus of claim 17, wherein the execution resources include one or more processor cores capable of executing multiple shreds concurrently.

23. (Currently Amended) The apparatus of claim 17, ~~further comprising:~~ wherein the shared register is ~~one or more registers~~ to hold at least a portion of a state shared among the first shred and the second shred. ~~one or more other shreds.~~

24. (Currently Amended) The apparatus of claim 17, wherein the first shred and the second shred ~~one or more other shreds~~ share a current privilege level and share a common address translation.

25. (Currently Amended) The apparatus of claim 17, further comprising logic to execute a user-level instruction to create the first shred.

26. (Currently Amended) The apparatus of claim 17, wherein the shared register comprises a first register to be utilized as a register semaphore to synchronize data between the first and the second shred ~~further comprising a mechanism to perform communication between the shred and the one or more other shreds~~.

27. (Currently Amended) The apparatus of claim 17, further comprising a group of registers to ~~sharing~~ share a system state among the first shred and the second shred ~~one or more other shreds~~.

28. (Previously Presented) The apparatus of claim 17 ~~26~~, further comprising a group of registers, which does not include the shared register, to hold a private portion of a state to be associated with the first shred. ~~wherein the mechanism further comprises one or more shared registers~~.

29. - 31 (Canceled).

32. (Currently Amended) The apparatus of claim 17, further comprising: a user-level exception mechanism to report an exception to the first shred.

33. (Previously Presented) The apparatus of claim 17, further comprising: an exception mechanism to report an exception to an operating system.

34. (Currently Amended) The apparatus of claim 32, further comprising:

a mechanism to detect a first exception associated with the first shred and a second exception associated with the second shred ~~multiple exceptions, each exception associated with a different one of a plurality of concurrently-executing shreds, where the plurality includes the shred and the one or more other shreds~~;

wherein the exception mechanism includes a prioritizer to prioritize the first and the second exceptions;

and wherein the exception mechanism is further to report only one of the prioritized first and second exceptions at a time to the operating system.

35. (Currently Amended) An article of manufacture, comprising:

a machine-accessible medium including data that, when accessed by a machine, cause the machine to perform operations comprising,

receiving user-level programming instructions to execute a plurality of threads of execution, wherein ~~that~~ each of the plurality of threads of execution are to be associated with a private state and to share ~~shares~~ a system state with an OS-generated thread; and

executing the plurality of threads of execution concurrently on multiple instruction sequencers of a processor in the machine.

36. (Currently Amended) The article of manufacture of claim 35, wherein ~~the operations further comprise: maintaining a private state for each shred of the plurality of shreds, wherein~~ the private state is associated with at least one of a plurality of registers including general purpose registers, floating point registers, MMX registers, segment registers, a flags register, an instruction pointer, control and status registers, SSE registers, and a MXCSR register.

37. (Currently Amended) The article of manufacture of claim 35, wherein ~~the machine-accessible medium further includes data that causes the machine to perform operations comprising sharing a first state among the plurality of shreds, while maintaining a second state privately among an additional shred associated with a thread that is not associated with the plurality of shreds, wherein the first state~~ the private state includes a private portion of an application state, and wherein the plurality of threads of execution are also to share a shared portion of the application state, the shared portion of the application state to be ~~is~~ associated with at least one of a plurality of registers including a control register, a flags register, memory management registers, a local descriptor table register, a task register, debug registers, model specific registers, shared registers, and shred control registers.

38. (Currently Amended) The article of manufacture of claim 35, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising: sharing a state among the plurality of threads of execution ~~shreds~~; and storing the state in one or more registers.

39. (Currently Amended) The article of manufacture of claim 35, wherein the plurality of threads of execution ~~shreds~~ share a current privilege level and share a common address translation.

40. (Currently Amended) The article of manufacture of claim 35, wherein the ~~one or more~~ user-level programming instructions include an instruction to create one or more of the plurality of threads of execution ~~shreds~~.

41. (Currently Amended) The article of manufacture of claim 35, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising communicating among the plurality of threads of execution ~~shreds~~.

42. (Currently Amended) The article of manufacture of claim 35, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising sharing a system state among the plurality of threads of execution ~~shreds~~.

43. (Currently Amended) The article of manufacture of claim 35, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising communicating between the plurality of threads of execution ~~shreds~~ via one or more shared registers.

44. (Currently Amended) The article of manufacture of claim 35, wherein an application program controls the plurality of threads of execution ~~shreds~~ directly, including scheduling of the plurality of threads of execution ~~shreds~~, and wherein an operating system executed by the machine schedules ~~one or more~~ the OS-generated thread ~~threads~~.

45. (Currently Amended) The article of manufacture of claim 35, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising: associating the plurality of threads of execution ~~shreds~~ with the OS-generated thread ~~a thread~~; and suspending the plurality of threads of execution ~~shreds~~ belonging to the OS generated thread when a context switch request is received through a ~~single~~ one of the plurality of threads of execution ~~shreds~~.

46. (Currently Amended) The article of manufacture of claim 45, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising: storing one or more threads of execution ~~shreds~~ states associated with the plurality of threads of execution ~~shreds~~ when the context switch request is received.

47. (Currently Amended) The article of manufacture of claim 35, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising: reporting one or more exceptions to a first ~~shred~~ thread of the plurality of threads of execution ~~shreds~~.

48. (Previously Presented) The article of manufacture of claim 47, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising: reporting the one or more exceptions from an application program; and determining whether to report the one or more exceptions to an operating system.

49. (Currently Amended) The article of manufacture of claim 48, wherein the machine-accessible medium further includes data that causes the machine to perform operations comprising prioritized-reporting of the one or more exceptions to the operating system; comprising

receiving the one or more exceptions concurrently via different threads ~~shreds~~ of the plurality of threads of execution shreds; and

servicing one of the one or more exceptions according to the prioritized-reporting while suspending exception processing of other exceptions of the one or more exceptions.

50. (Currently Amended) The article of manufacture of claim 35, wherein the plurality of threads of execution ~~shreds~~ perform input/output (I/O) functions and computation functions.

51. (Currently Amended) A system, comprising:

a microprocessor implementing an instruction set architecture (ISA), the microprocessor capable of executing multiple OS-generated threads, wherein the microprocessor is also capable of concurrently executing multiple shared resource threads (shreds) ~~concurrent shreds~~ associated with one of the OS-generated threads; and

a memory to store one or more instructions of the ISA~~;~~ ~~wherein the ISA includes one or more instructions~~ that is to allow user-level multithreading operations.

52. Canceled.

53. Canceled.

54. Canceled.

55. (Previously amended) A system, comprising:

a microprocessor, including a plurality of user-level multithreading registers, wherein the registers are addressable by one or more user-level instructions in each of a plurality of user-level threads and are to support communication among the user-level threads; and

memory coupled to the microprocessor, the memory to store the one or more user-level instructions;

wherein the microprocessor is further to execute the user-level threads concurrently.

56. (Original) The system of claim 55, wherein the plurality of user-level multithreading registers further comprises a plurality of shared shred registers to facilitate communication between a plurality of shreds and to facilitate synchronization between the plurality of shreds.

57. (Original) The system of claim 56, wherein the plurality of user-level multithreading registers further comprises a plurality of shred control registers to manage the plurality of shreds.

58. (Previously Presented) The system of claim 57, wherein the microprocessor:

receives programming instructions to execute one or more shreds in accordance with the ISA;

configures one or more instruction sequencers via the ISA; and

executes the one or more shreds concurrently.

59. (Previously presented) The apparatus of claim 32, wherein:

the user-level exception mechanism is further to vector to a fixed location in order to allow the shred to service to the exception.

60. (Previously presented) The apparatus of claim 32, wherein:
the plurality of instructions further include a system call instruction to explicitly invoke an operating system to service to the exception.

61. (Currently Amended) The apparatus of claim [[33]]34, wherein said prioritizer employs a round-robin scheme.

62. (Previously presented) The article of manufacture of claim 35, wherein the one or more user-level programming instructions include an instruction to destroy one or more of the plurality of shreds.

63. (Previously presented) The system of claim 51, wherein the one or more instructions include an instruction to create a shred without intervention of an operating system.

64. (Previously presented) The system of claim 51, wherein the one or more instructions include an instruction to destroy a shred without intervention of an operating system.

65. (Previously presented) The system of claim 51, wherein:
the user-level multi-threading operations include concurrent execution of two or more shreds associated with the same thread.

66. (New) The system of Claim 55, wherein the memory is from a plurality of memory devices including DRAM, flash, and EEPROM.

67. (Currently Amended) An apparatus ~~processor~~ comprising:

a processor capable of user-level multithreading including:

a first group of resources to hold a per shared resource thread ("shred") application state for a first shred to be created by a first user-level instruction;

a second group of resources to hold a per shred application state for a second shred to be created by a second user-level instruction; and

a third group of resources to be shared by the first shred and the second shred to hold a shared application state, the shared application state to be shared by a privileged-level software entity created thread; and

execution resources to concurrently execute the first shred and the second shred.

~~execution resources to execute instructions of an instruction set architecture (ISA), the execution resources to execute multiple concurrent shared resource threads ("shreds") that share application state;~~

~~wherein the ISA includes one or more non-privileged instructions to allow multithreading operations.~~

68. (Currently Amended) The ~~processor~~ apparatus of Claim 67, wherein: the first group of resources, the second group of resources, and the third group of resources include registers, and wherein the first user-level instruction and the second user-level instruction are an Instruction Set Architecture instructions to create a shred, which are to be associated with a first user software entity and a second user software entity, respectively.

~~the one or more non-privileged instructions include an instruction to create a shared resource thread.~~

69. (New) The apparatus of Claim 68, further comprising a hardware re-namer to allocate the first group of registers as private registers and the third group of registers as shared registers based on a bit vector.

70. (New) The apparatus of Claim 67, wherein the application state to be shared by the privileged-level software entity thread is not shared with other privileged-level software entity threads, and wherein the privileged level software entity is an operating system (OS)..

71. (New) The apparatus of Claim 67, wherein the second group of resources is a copy of the first group of resources, and wherein the first group of resources includes a combination of resources selected from a group consisting of general registers, floating point registers, SSE registers, instruction pointer, and flags, and wherein the third group of resources includes a combination of resources selected from a group consisting of general registers, floating point registers, shared communication registers, flags, memory management registers, address translation tables, privilege levels, and control registers.